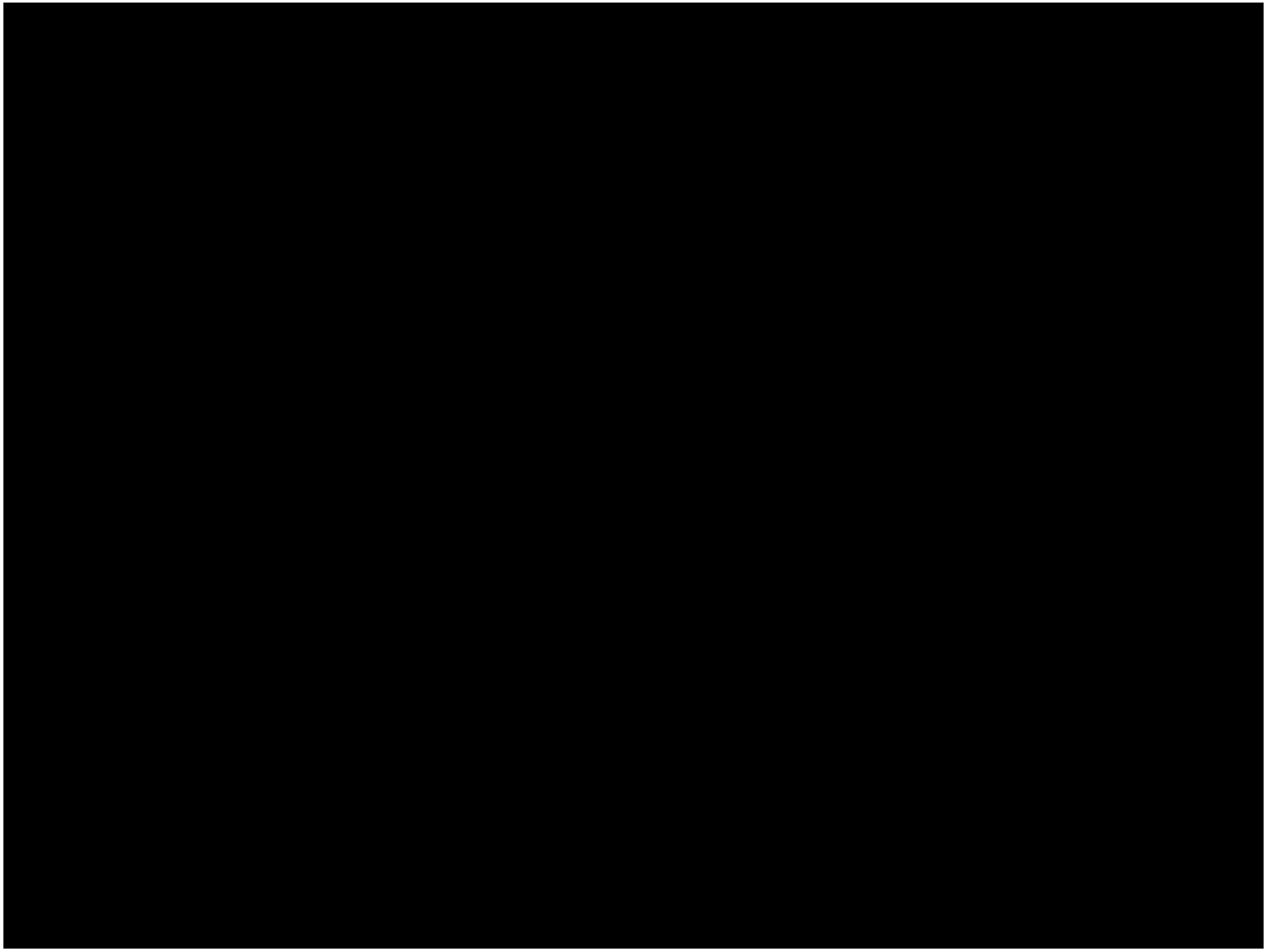

Productive Potato Sloth

An overview of our specification

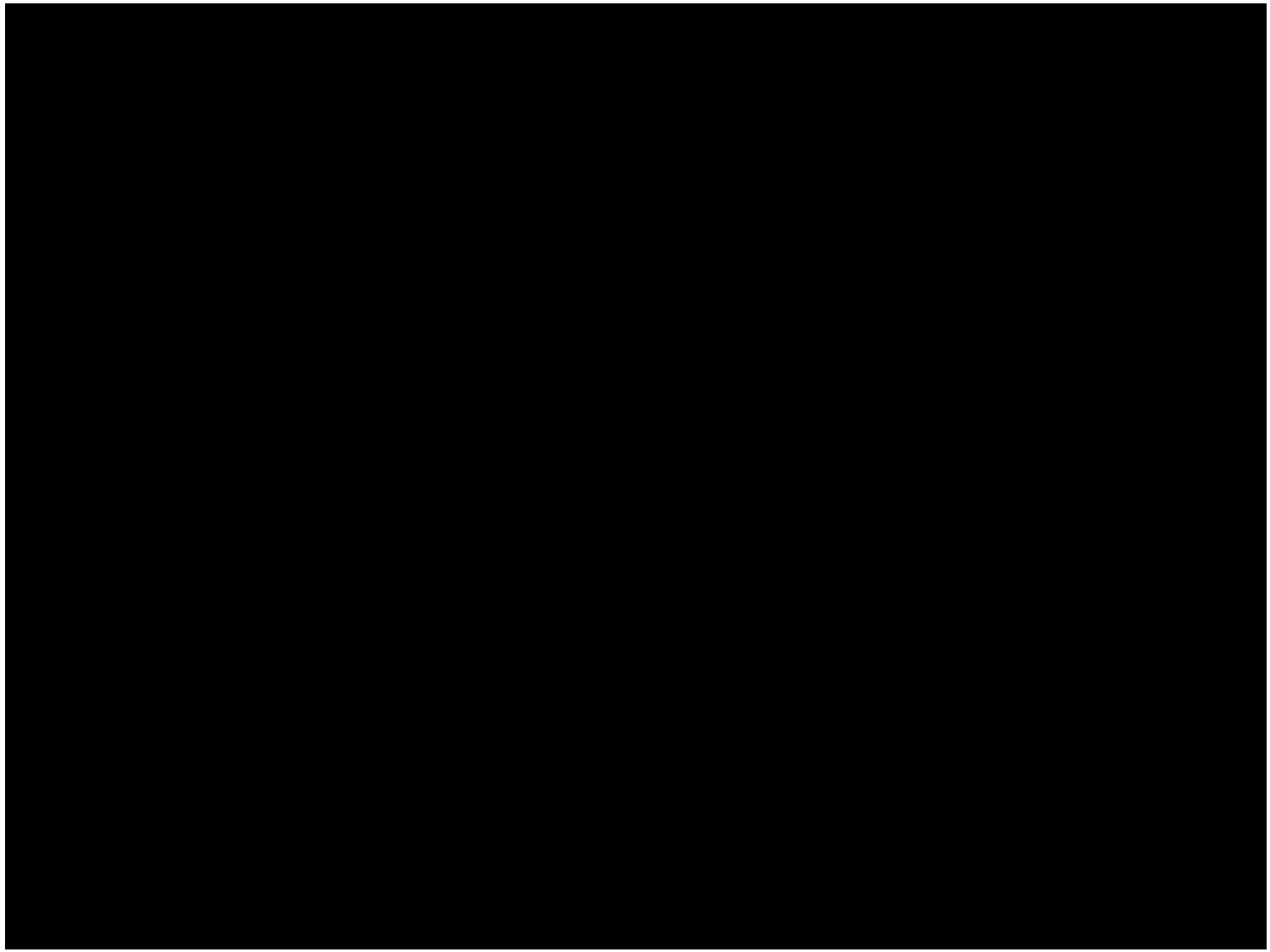
- productivity application
- users may:
  - create accounts and log in
  - create, view, and edit tasks
  - invite and accept invitations to collaborate on tasks
  - initiate timers
  - contribute to a chat room discussion on each task
  - create and view events
  - view usage statistics

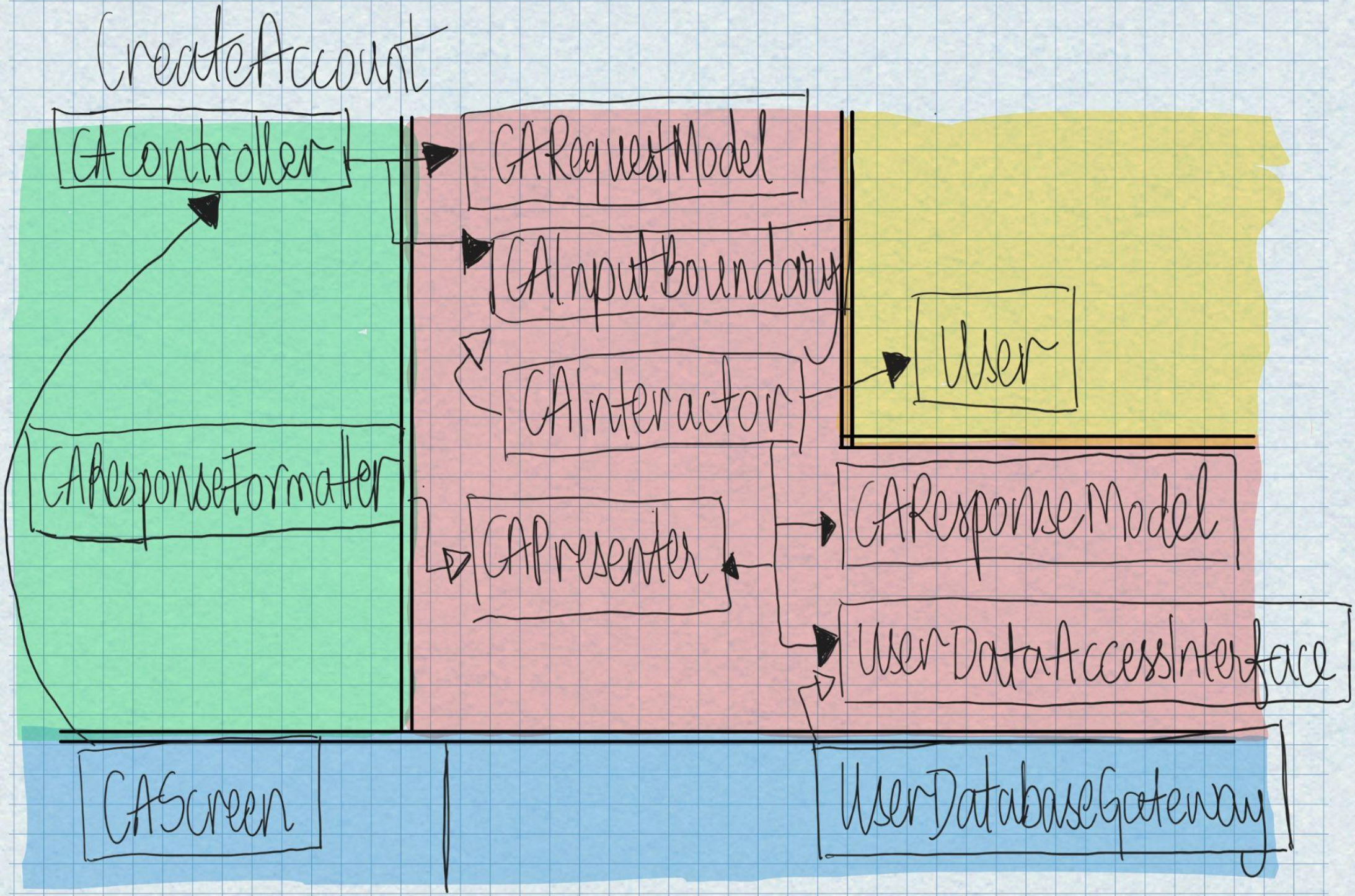
Logging In /
Creating an Account (User)

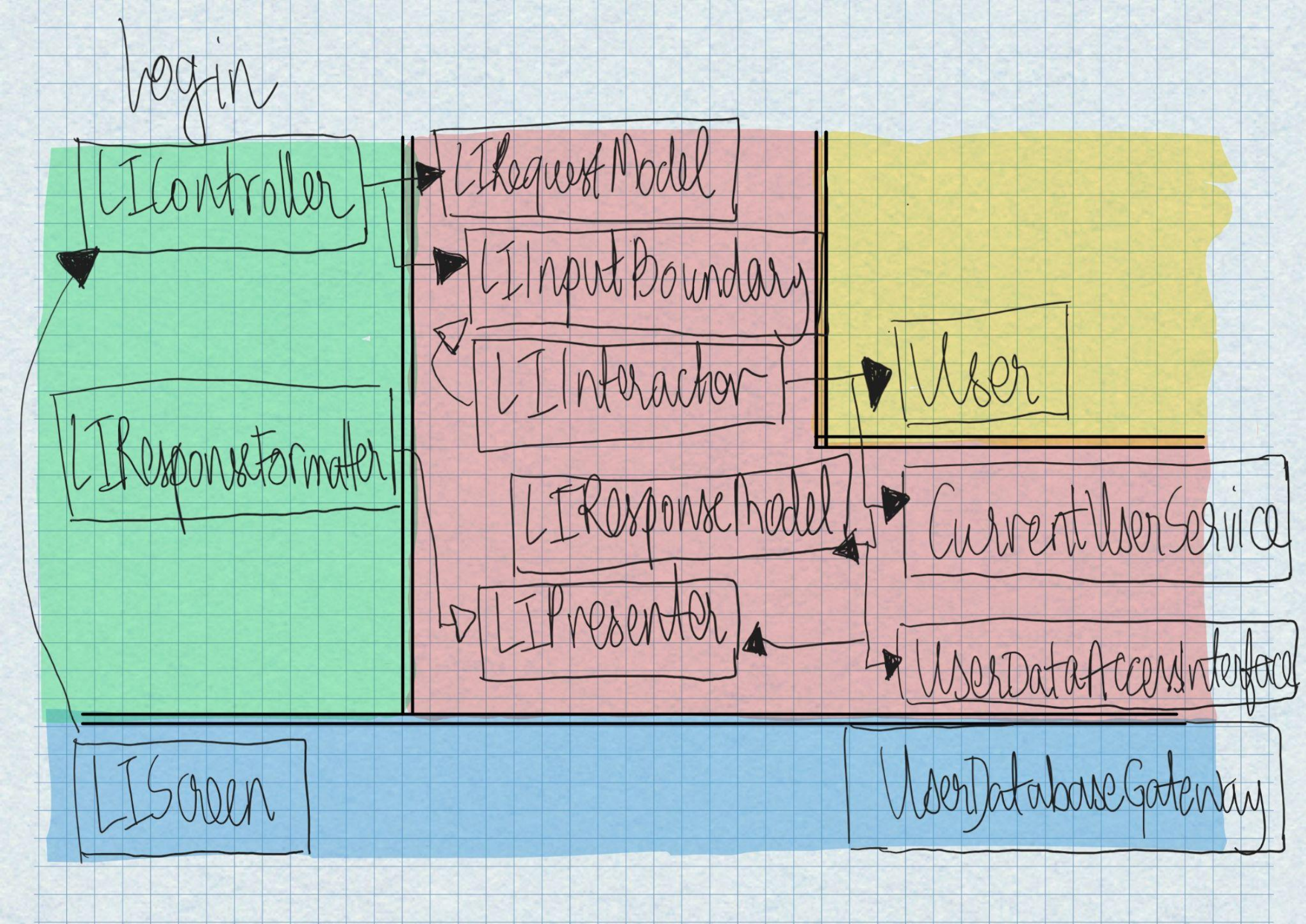
Create Account Demo!



Login Demo!




CreateAccount
CAController
CARequestModel
CAInputBoundary
CAInteractor
User
CAResponseFormatter
CAPresenter
CAResponseModel
UserDataAccessInterface
CAScreen
UserDatabaseGateway


login
LIController
LIRequestModel
LIInputBoundary
LIInteractor
User
LIResponseFormatter
LIResponseModel
CurrentUserService
LIPresenter
UserDataAccessInterface
LIScreen
UserDatabaseGateway

Do we appreciate Software Engineers enough?

CreateAccountInteractor

Responsibilities

- createAccount:
  - Alternatively, allows user to create login with unique username and password. If username exists, gives error that "this username is invalid."
  - Updates User.username and User.password

Collaborators

- User

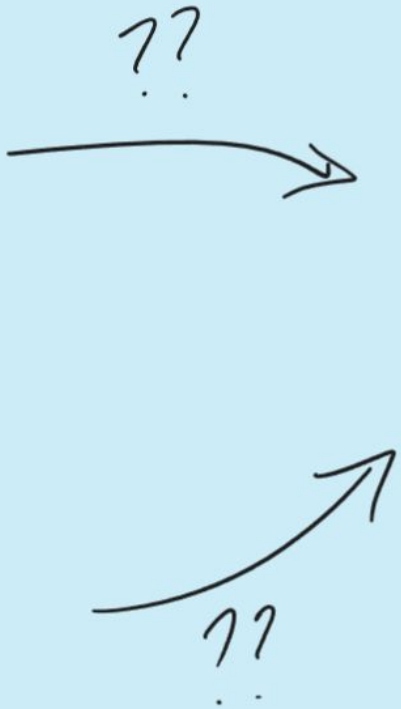
LoginPageInteractor

Responsibilities

- login:
  - Allows user to input a username and password.
- checkUsername:
  - Checks if username exists. If yes, confirms that password corresponds to username. If not, gives error that "username does not exist" or "password is incorrect".

Collaborators

- User

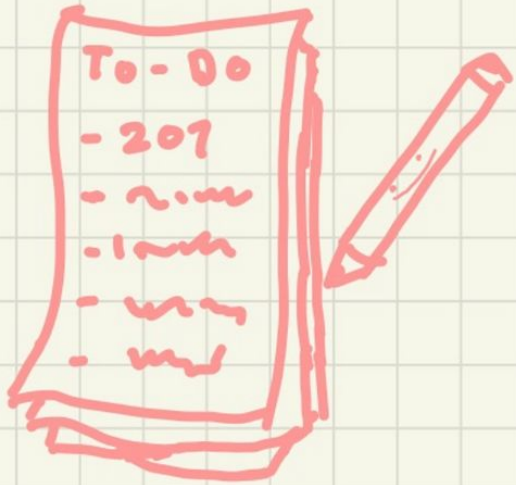


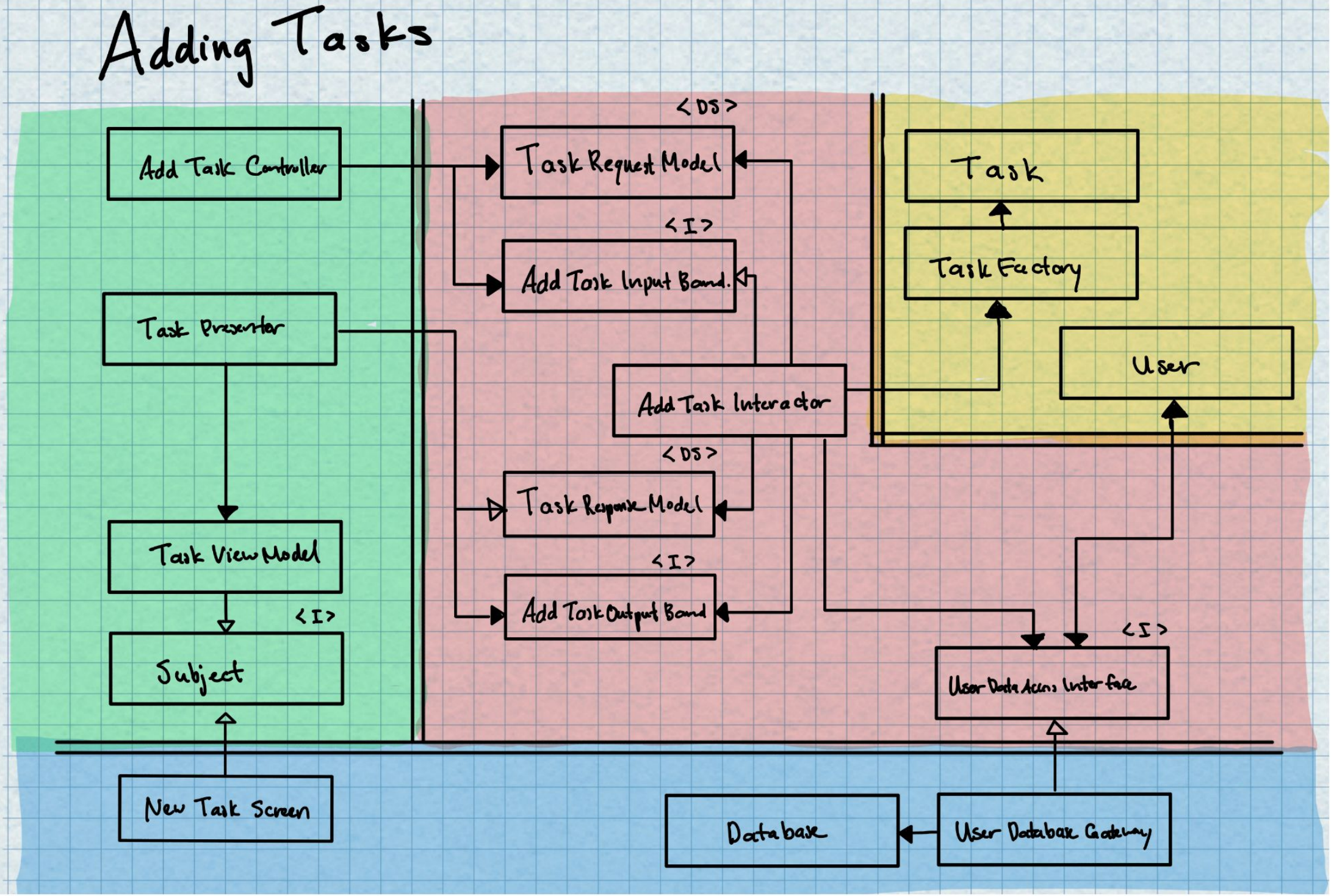
```java
}

Vishnu Nittoor
public static void main(String[] args) throws IOException {
    JFrame application  =  new JFrame( title: "Create Account");
    CardLayout cardLayout = new CardLayout();
    JPanel screens = new JPanel(cardLayout);
    application.add(screens);

    UserDataAccessInterface gateway = new UserDatabaseGateway( relativePath: "database/UserFile1.ser");
    CurrentUserService service = new CurrentUserService();

    System.out.println(gateway.getAll().get(0).getPassword());

    CreateAccountPresenter presenter = new CreateAccountResponseFormatter();

    CreateAccountInputBoundary createAccountInteractor = new CreateAccountInteractor(gateway, presenter);

    CreateAccountController createAccountController = new CreateAccountController(createAccountInteractor);

    CreateAccountScreen createAccountScreen = new CreateAccountScreen(createAccountController);

    screens.add(createAccountScreen,  constraints: "hello");

    cardLayout.show(screens,  name: "create account");
    application.pack();
    application.setVisible(true);


}
}
```

Task List

Features!

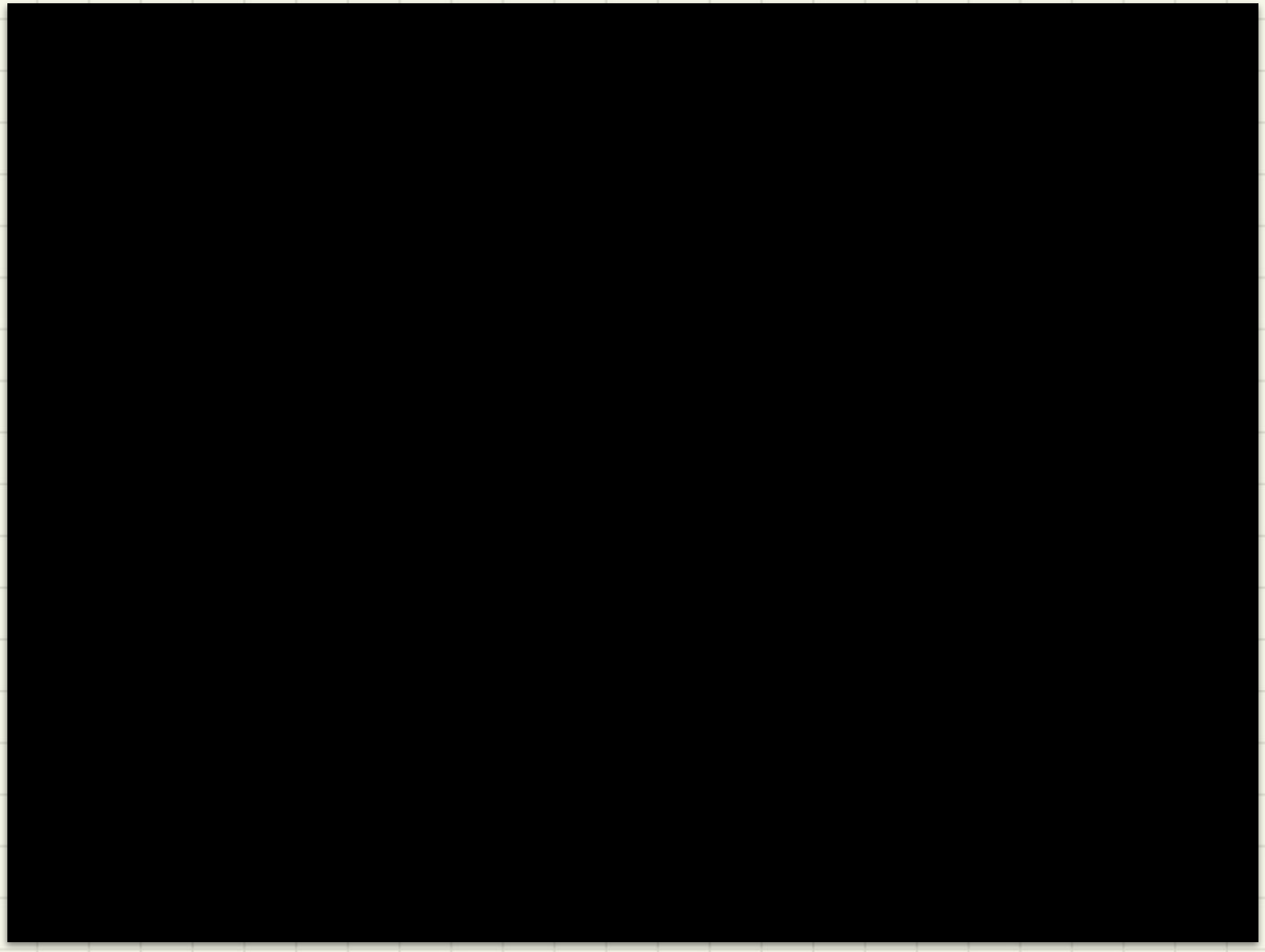
- Add + Delete Tasks
- Task Descriptions
- Checkboxes!
- Creating + Adding custom Tags
- Events
- Collaborators
- Chatrooms!




Adding Tasks
Add Task Controller
<DS>
Task Request Model
Task
<I>
Add Task Input Bound.
Task Factory
Task Presenter
User
Add Task Interactor
<DS>
Task Response Model
Task View Model
<I>
Add Task Output Bound
<I>
Subject
User Data Access Interface
New Task Screen
Database
User Database Gateway

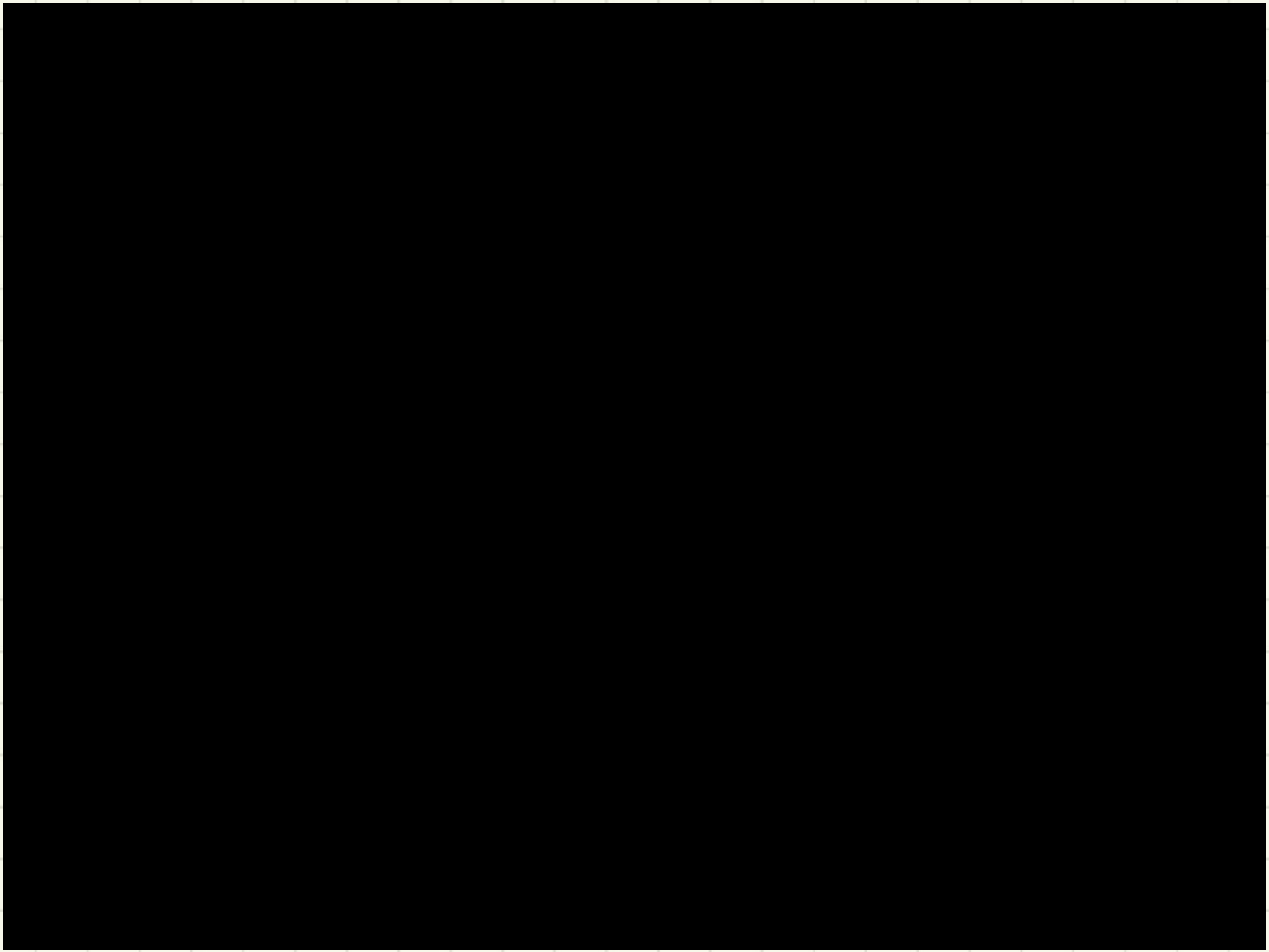
Feature Demo (!!)

Adding a Task



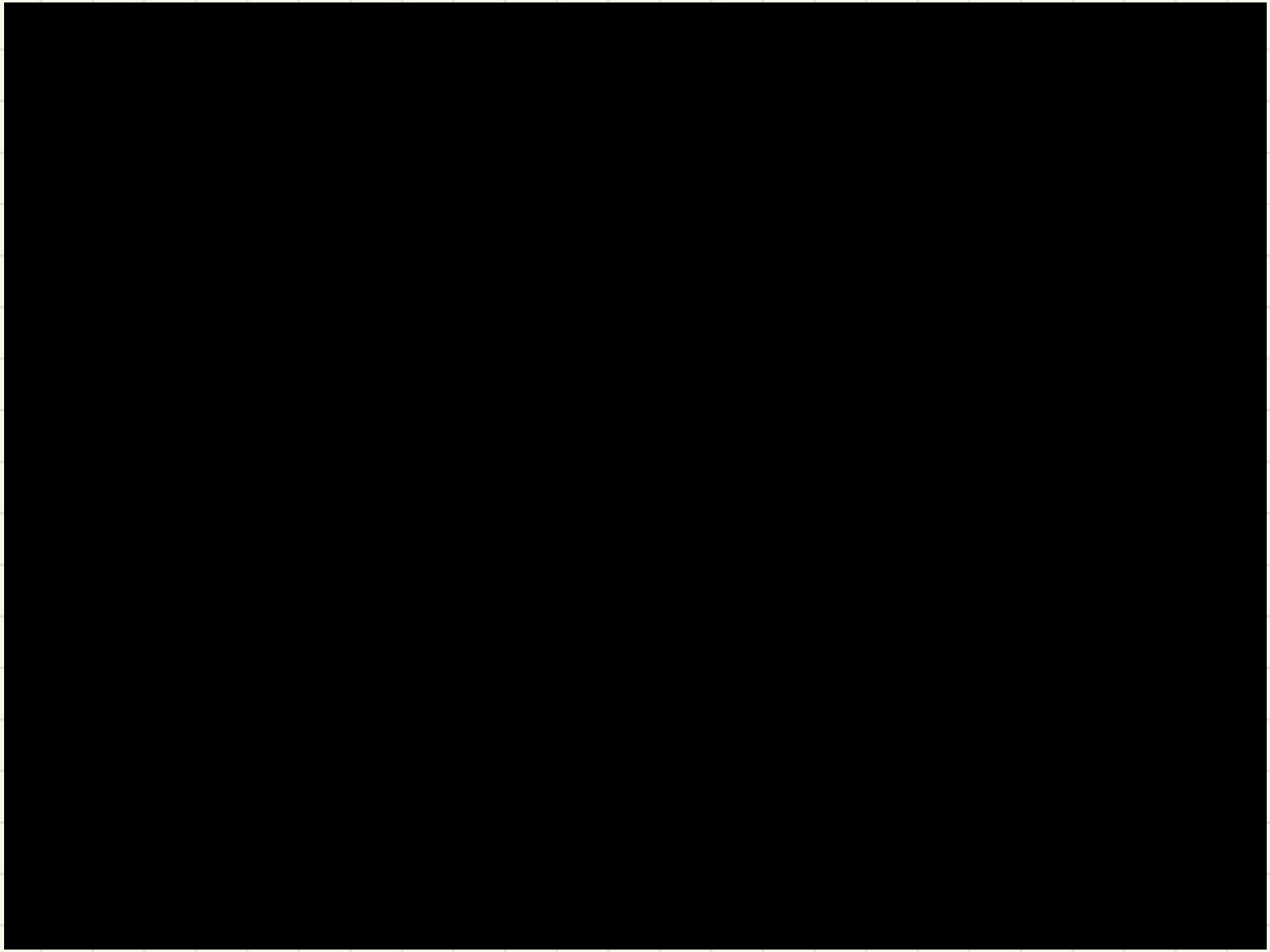
Feature Demo (!!)

Deleting a Task



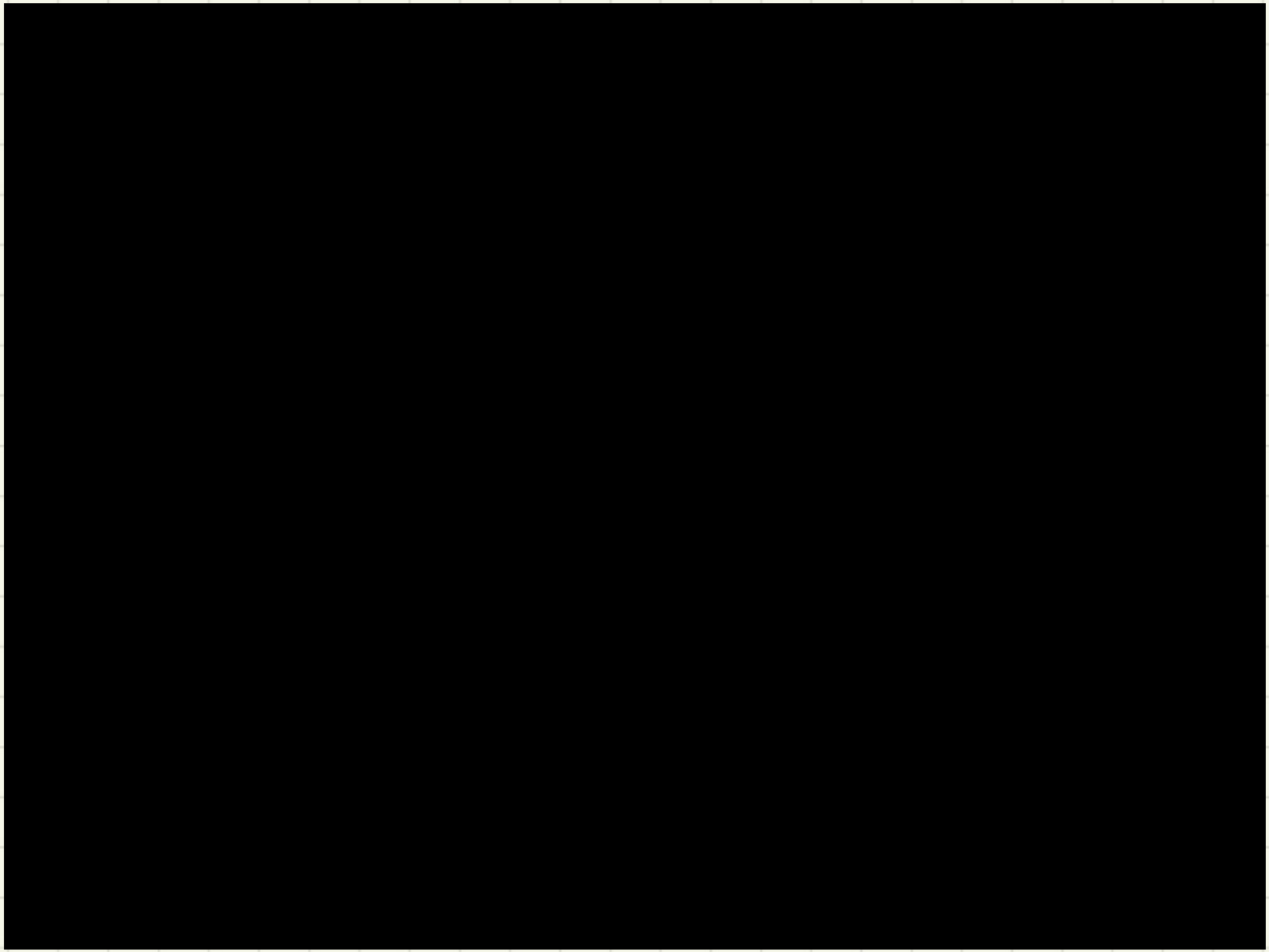
Feature Demo (!!)

Dynamic updating

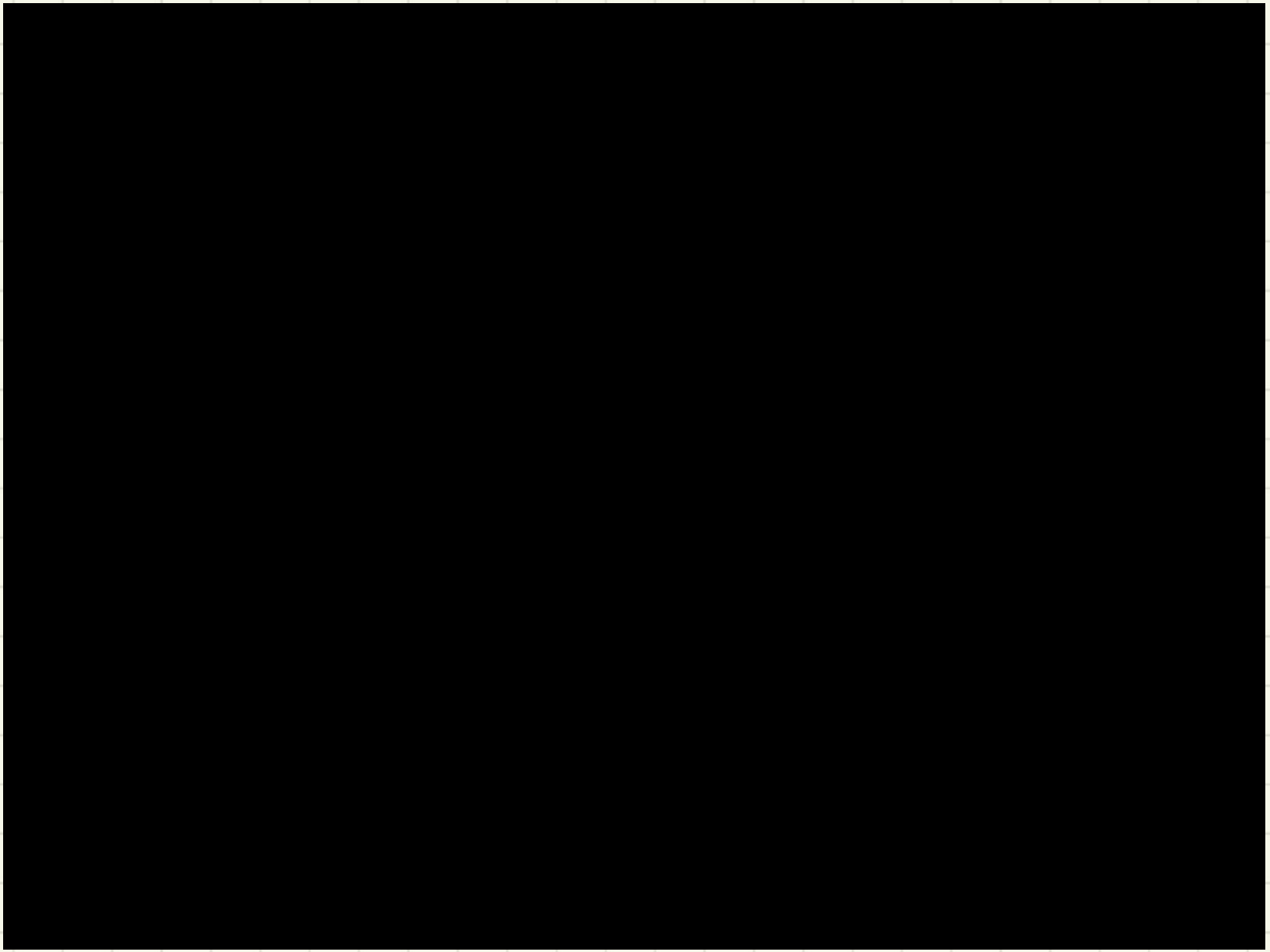


Feature Demo (!!)

Editing a Task



Feature Demo (!!)
ERROR!!!



Issue !!

- How will my screens reflect any changes to tasks ?

Solutions (?)

- Constantly check for updates

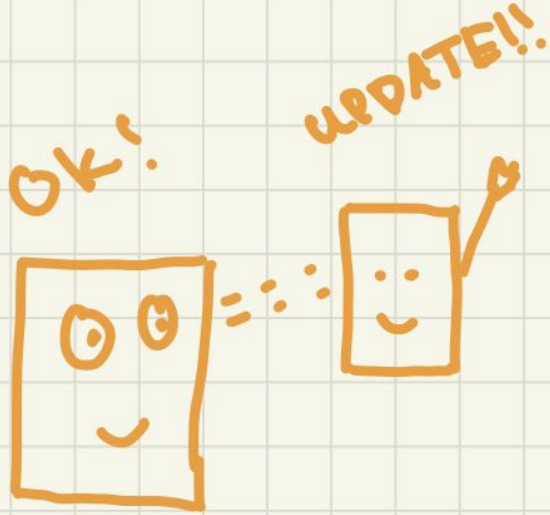
X HORRIBLE Design

- Observer pattern

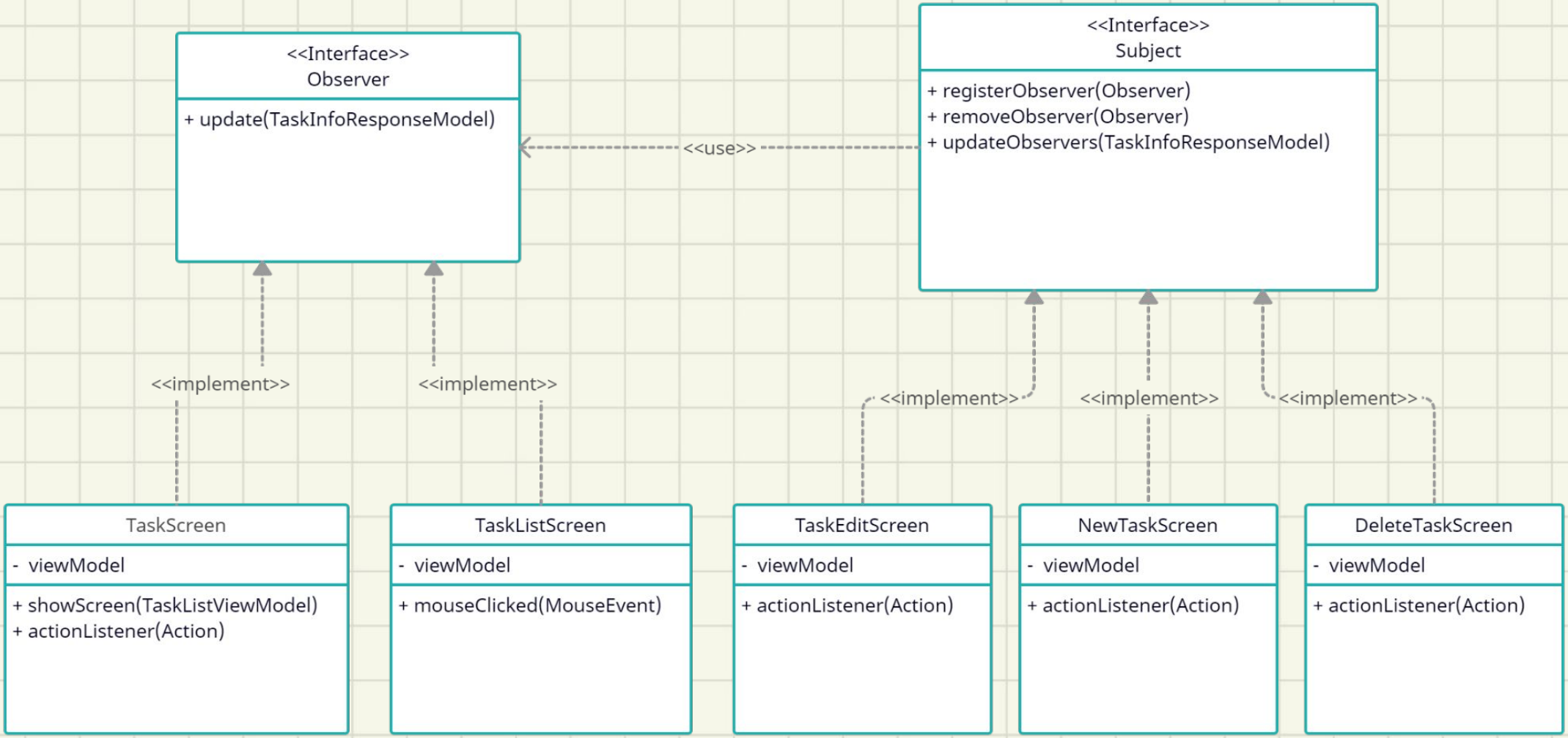
✓ Clean, open for extension

↑

SOLID !

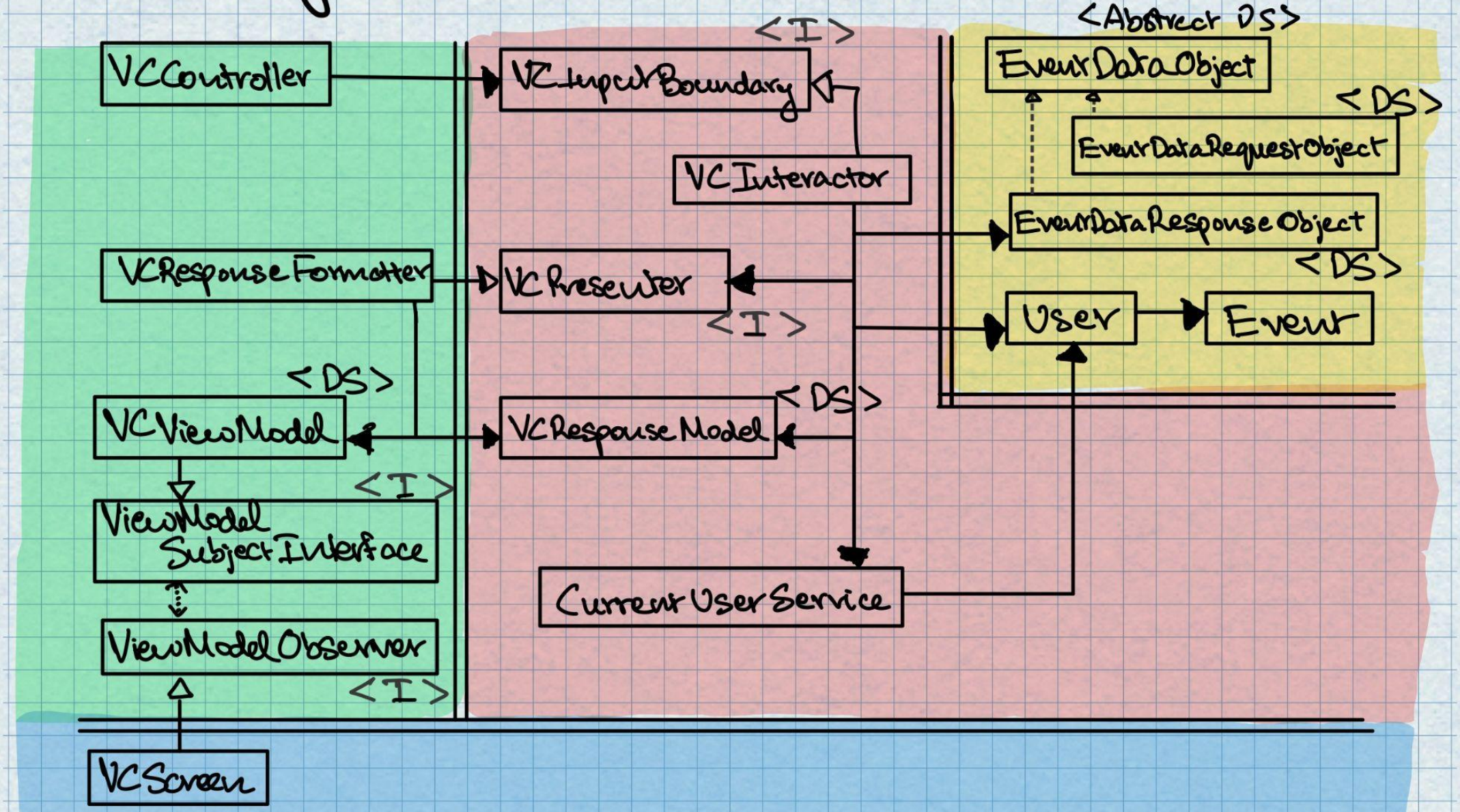


Observer Design

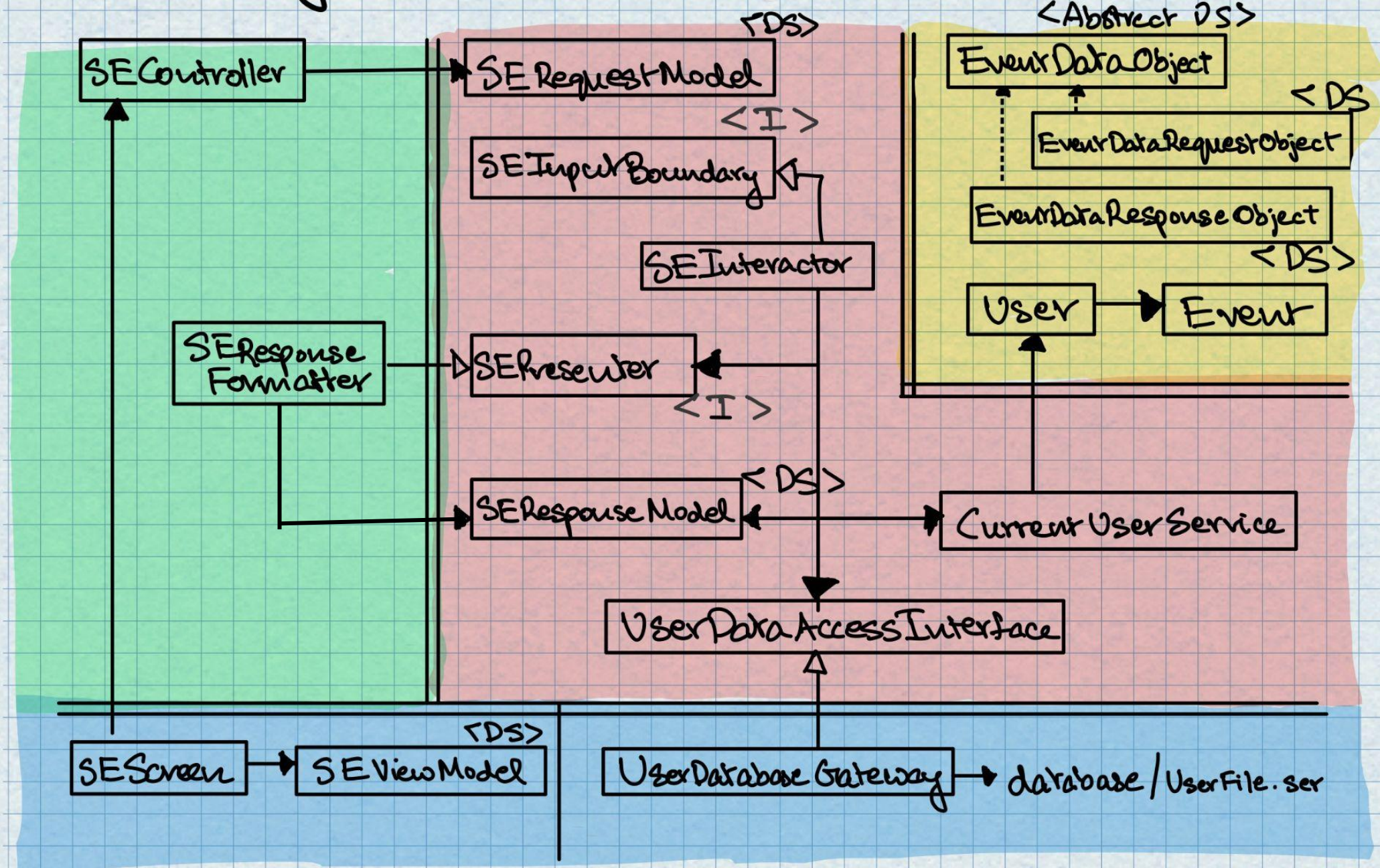


Viewing and scheduling events

Viewing Events



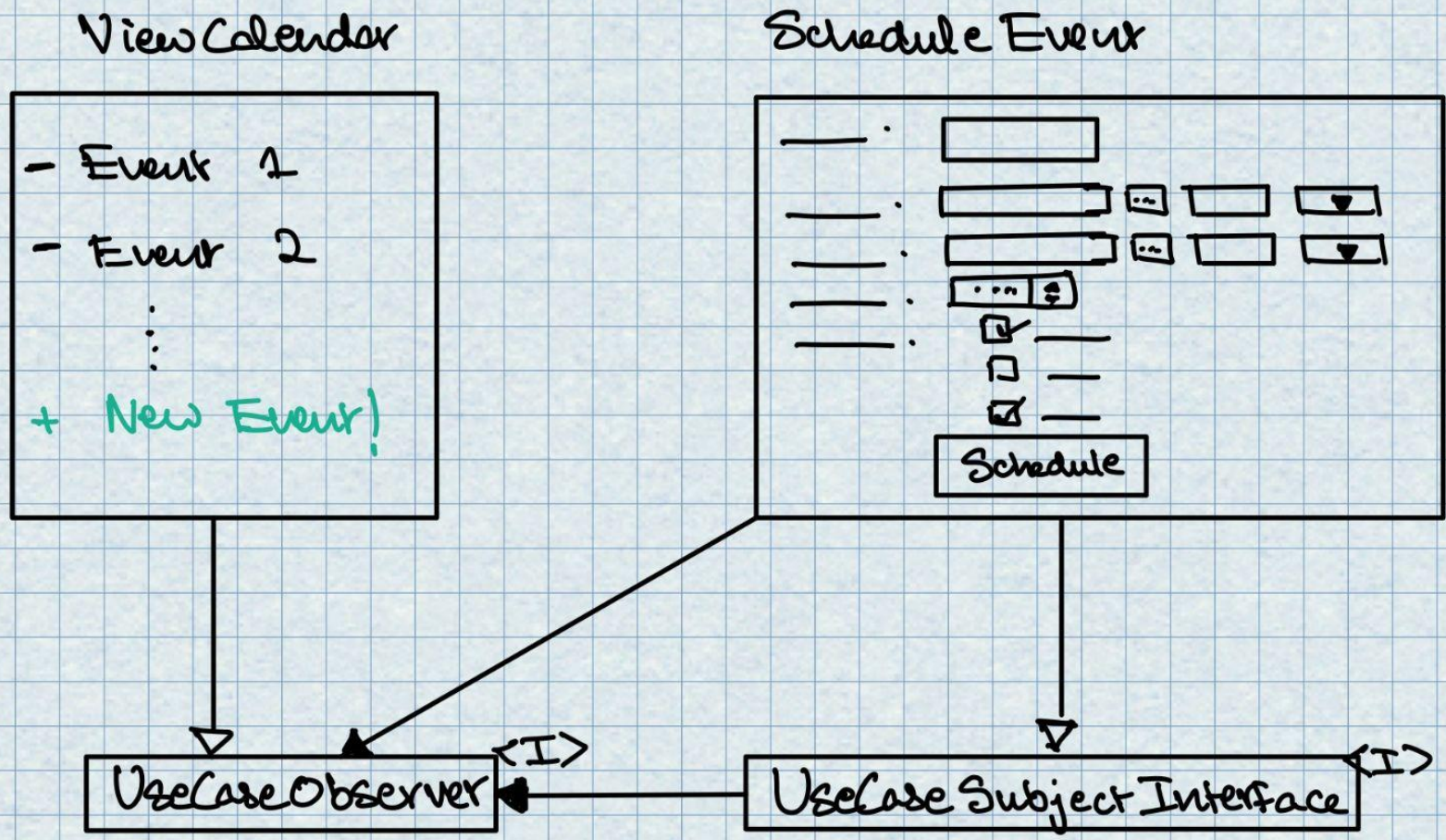
Scheduling Events



"I want to view my events while I schedule them!"

– Disgruntled User Who Kind of Has a Point

Solution: Run the ViewCalendarScreen and ScheduleEventScreen parallelly, and use the Observer pattern to trigger updates.

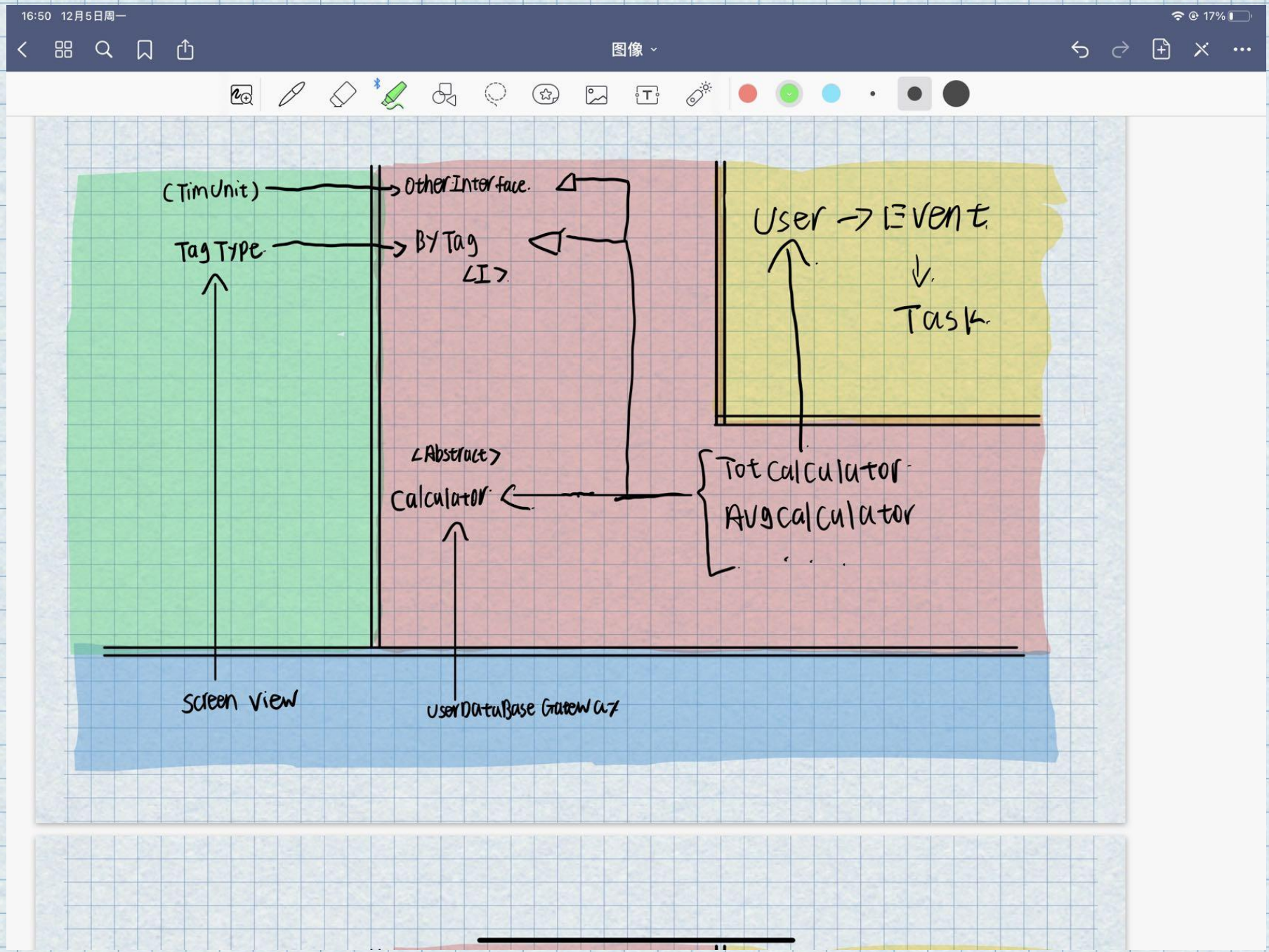


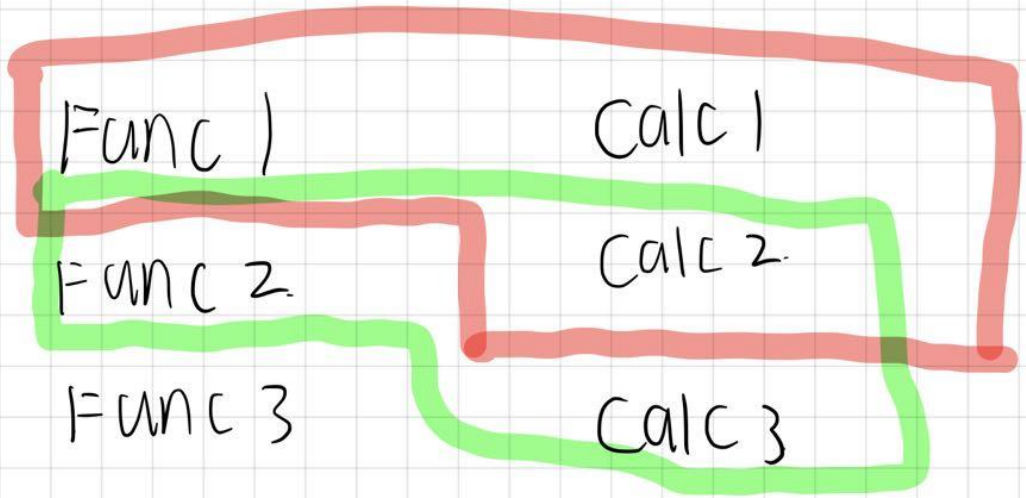
Demo!

Demo!


19:12
屏幕使用时间
Smeb#
每周
每天
屏幕使用时间
显示今天
11月27日 星期日
6小时15分钟
10小时
平均
5小时
0
日
一
二
三
四
五
六
60分钟
30分钟
0
0时
6时
12时
18时
娱乐
3小时30分钟
社交
1小时24分钟
信息与阅读
30分钟
更新于：今天 19:12
最常使用
显示类别
哔哩哔哩
3小时23分钟
微信
55分钟
豆瓣阅读
30分钟
QQ
13分钟

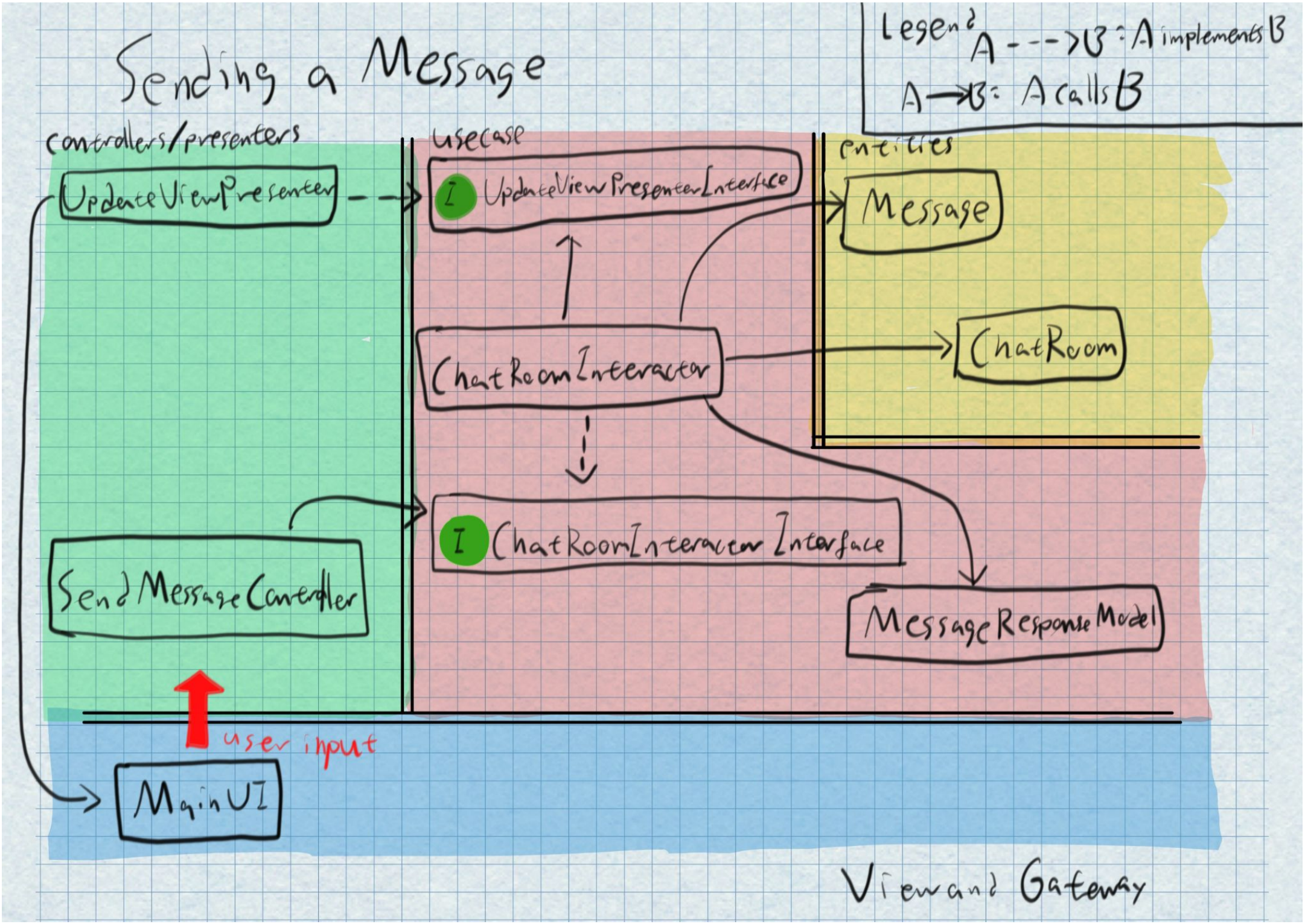
Demo !

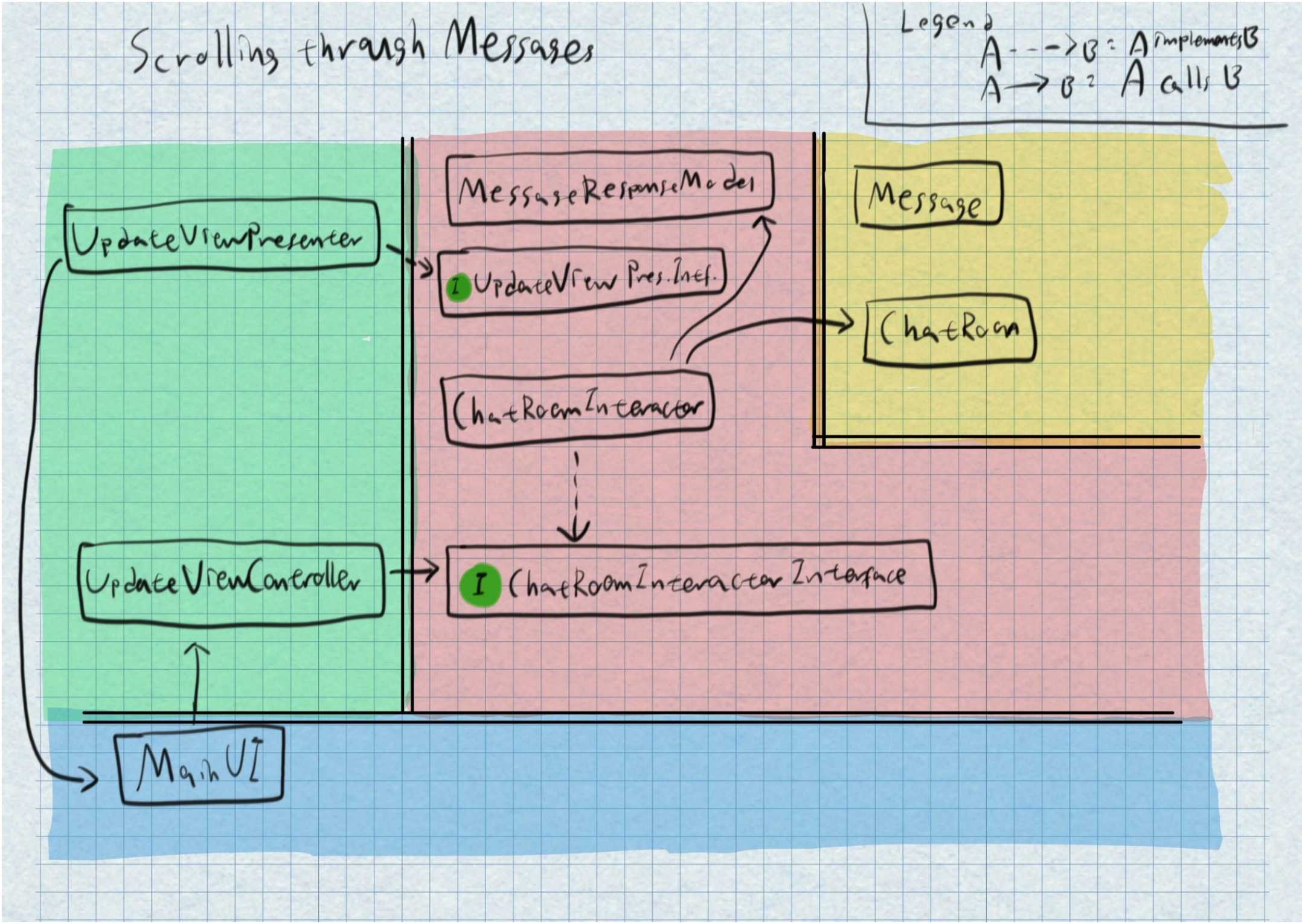

16:50 12月5日周一
17%
图像
(TimUnit)
OtherInterface.
TagType
BYTag
<I>
User → Event
Task
<Abstract>
Calculator
TotCalculator
AvgCalculator
. . .
Screen View
UserDataBaseGateway



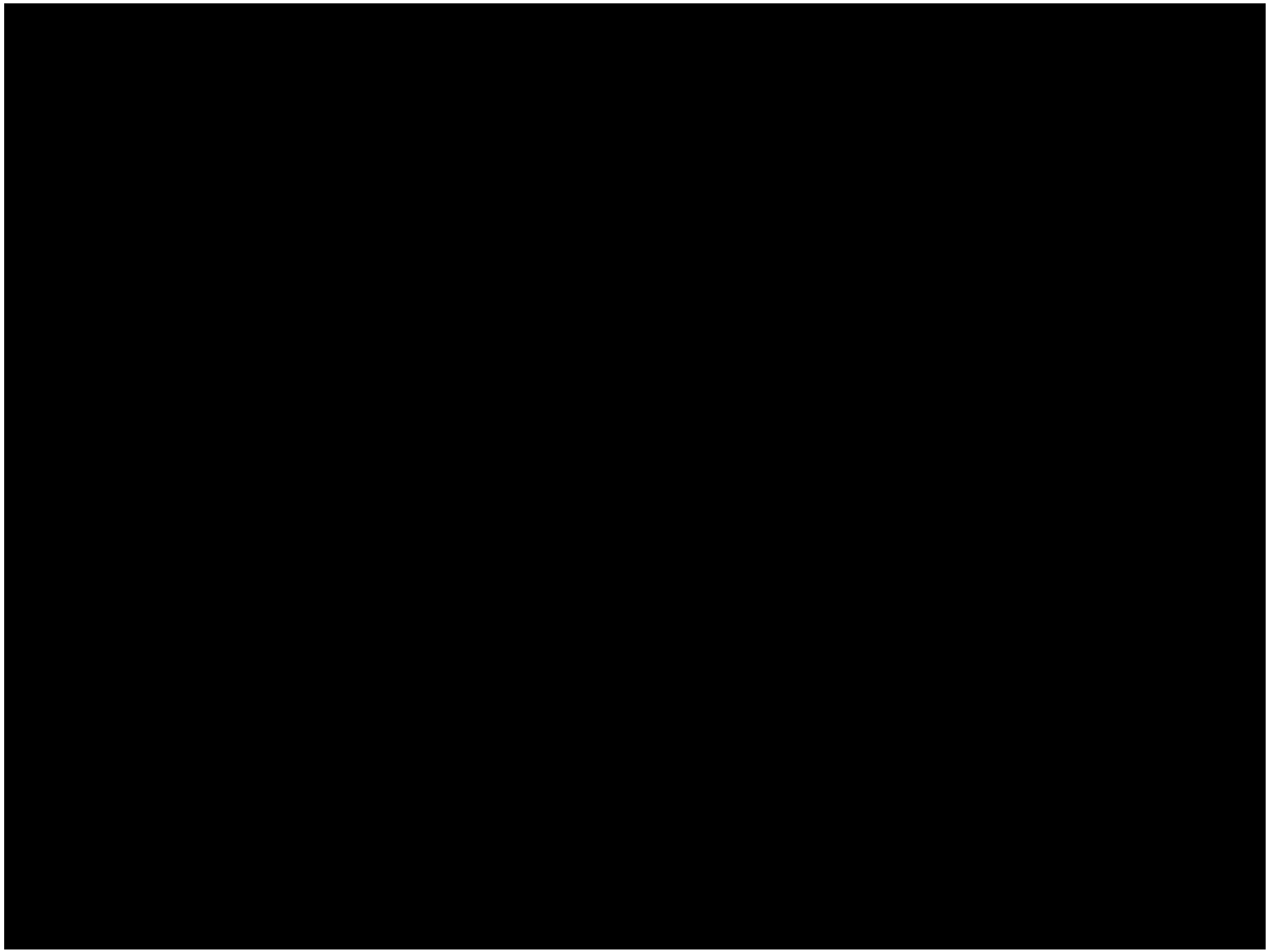
ChatRoom text feature

Dawei He


Sending a Message
Legend
A ---> B : A implements B
A → B : A calls B
controllers/presenters
usecase
entities
UpdateViewPresenter
I UpdateViewPresenterInterface
Message
ChatRoomInteracter
ChatRoom
I ChatRoomInteracterInterface
SendMessageController
MessageResponseModel
user input
MainUI
View and Gateway

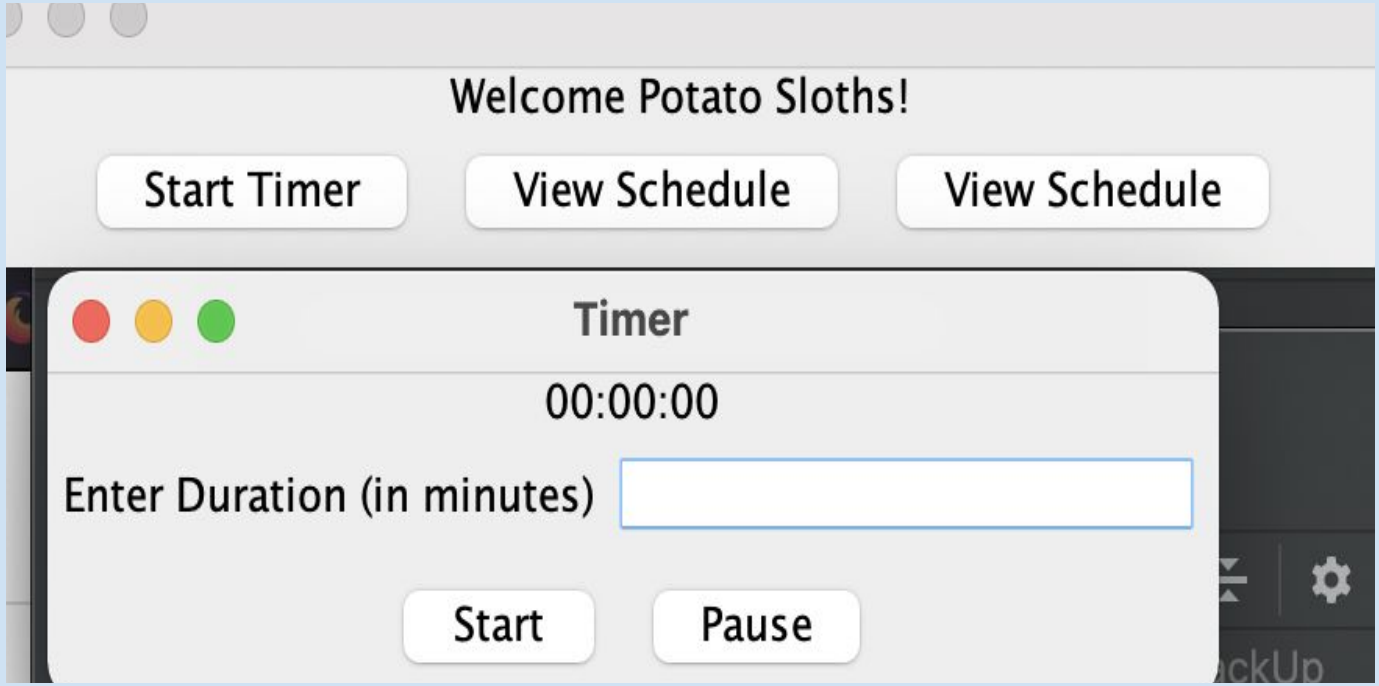

Scrolling through Messages
Legend
A - - -> B : A implements B
A -> B : A calls B
MessageResponseModel
Message
UpdateViewPresenter
I UpdateView Pres. Intf.
ChatRoom
ChatRoomInteractor
UpdateViewController
I ChatRoomInteractor Interface
MainUI

Demo

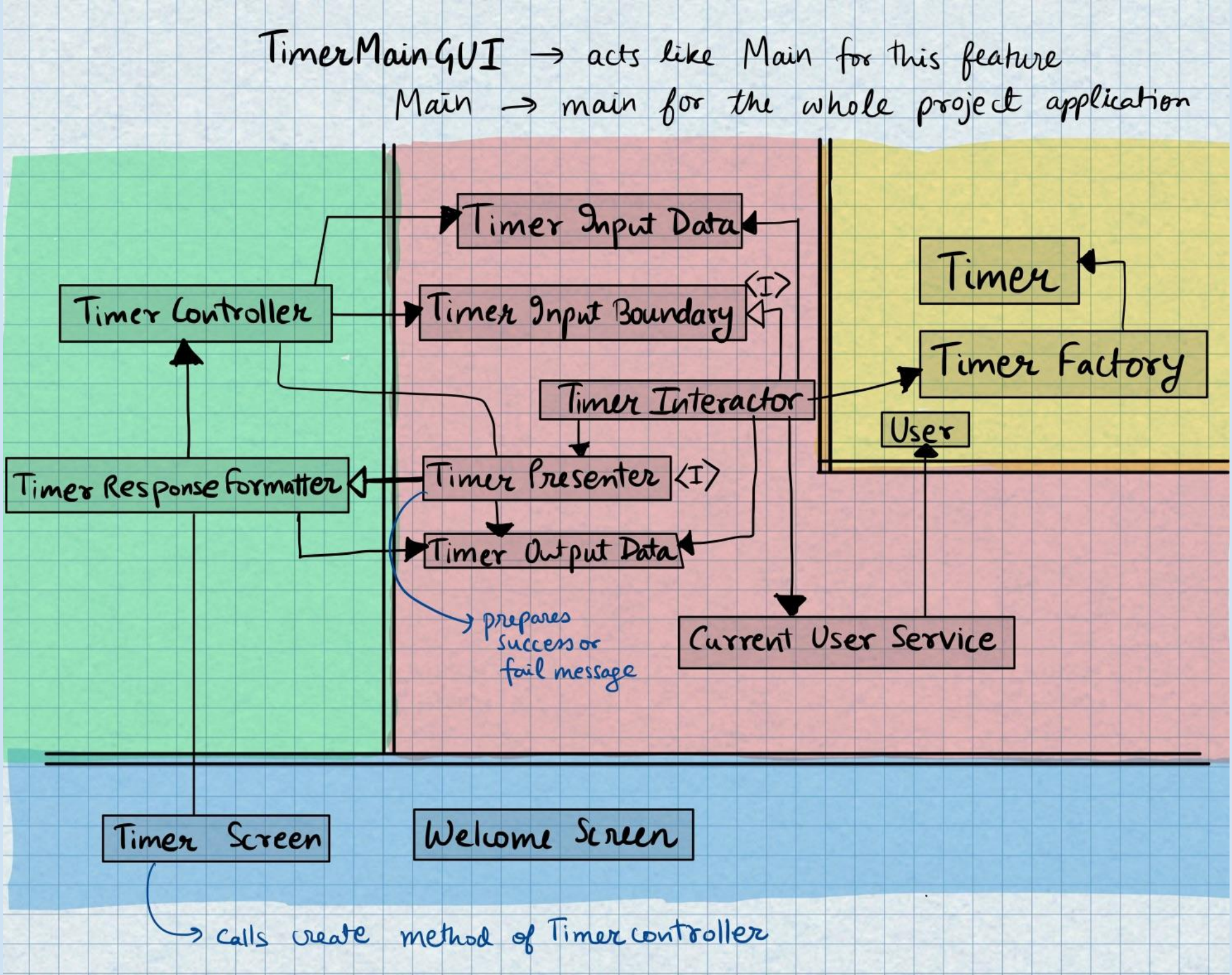


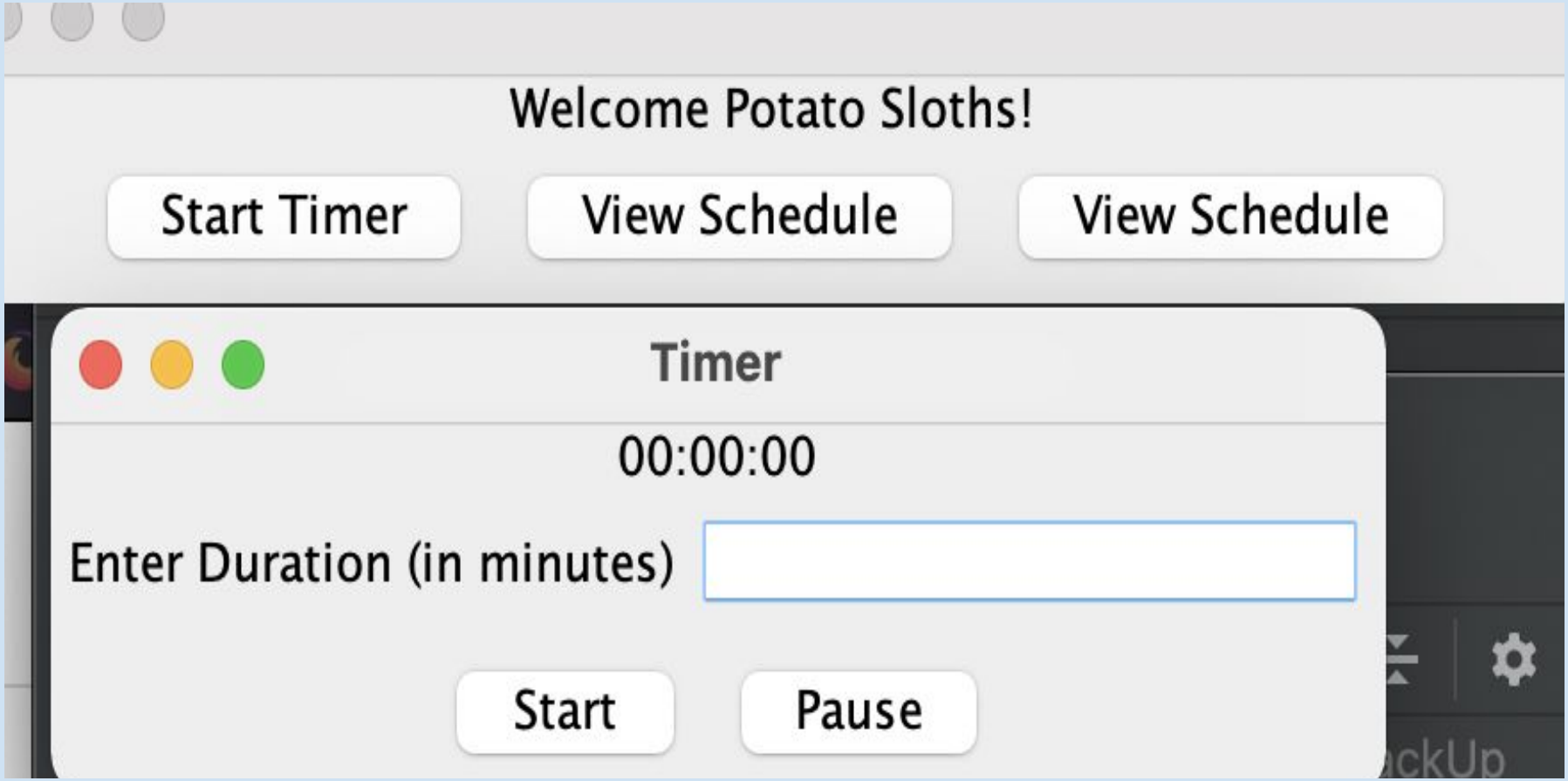
Challenges faced in development

Timer Feature



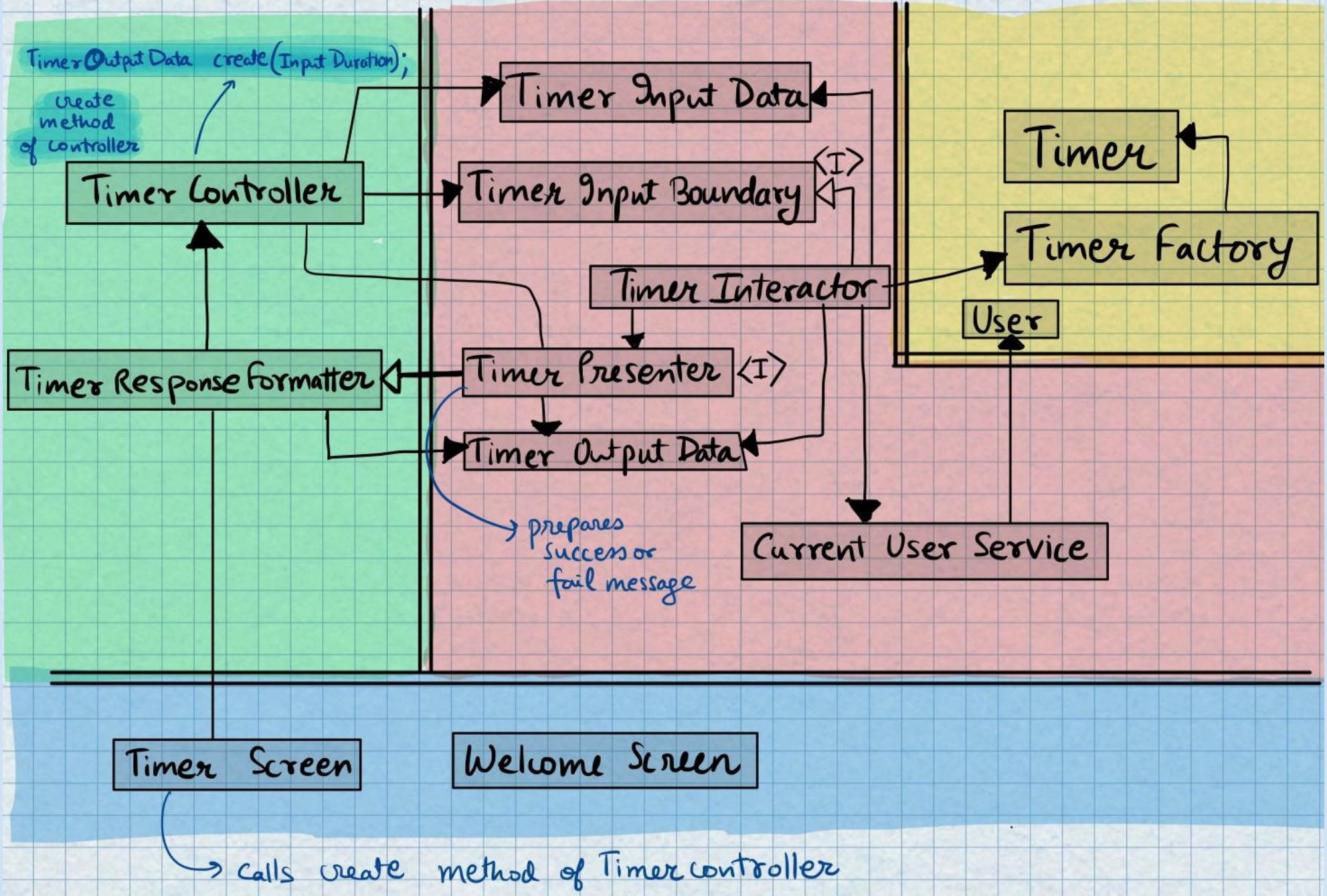
Timer Feature: Demo


TimerMainGUI → acts like Main for this feature
Main → main for the whole project application
Timer Input Data
Timer Controller
Timer Input Boundary
<I>
Timer
Timer Factory
Timer Interactor
User
Timer Response Formatter
Timer Presenter
<I>
Timer Output Data
prepares success or fail message
Current User Service
Timer Screen
Welcome Screen
calls create method of Timer controller

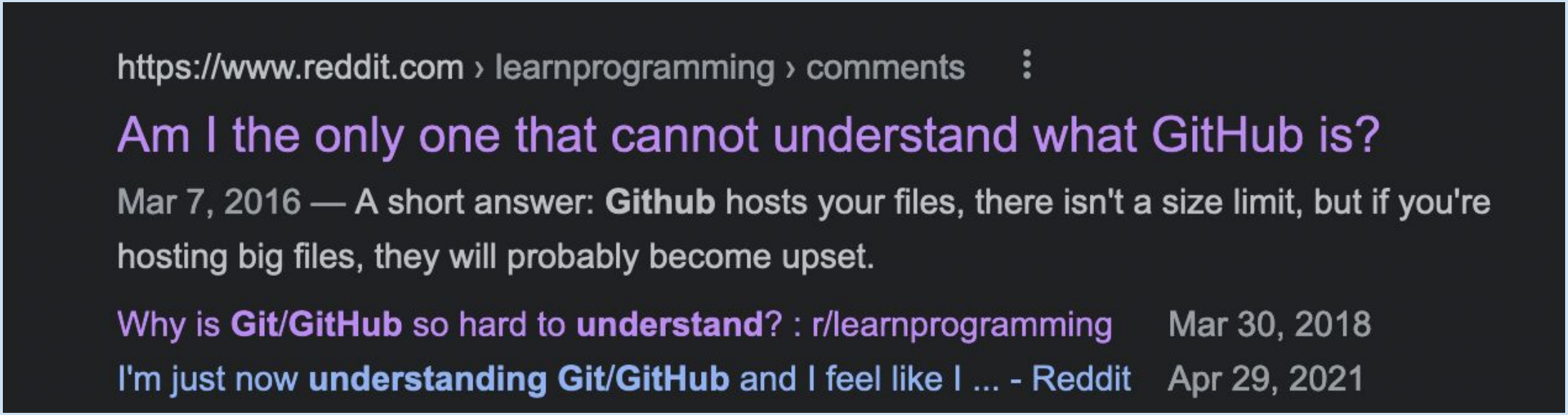

Welcome Potato Sloths!
Start Timer
View Schedule
View Schedule
Timer
00:00:00
Enter Duration (in minutes)
Start
Pause

TimerMainGUI → acts like Main for this feature

Main → main for the whole project application



Challenges:



Throughout the project, I found working with git and github far more difficult than the actual programming or design.

However, I feel much more confident working with git now than I did in the beginning, so there is a lot of progress!

I programmed something I can actually use myself.

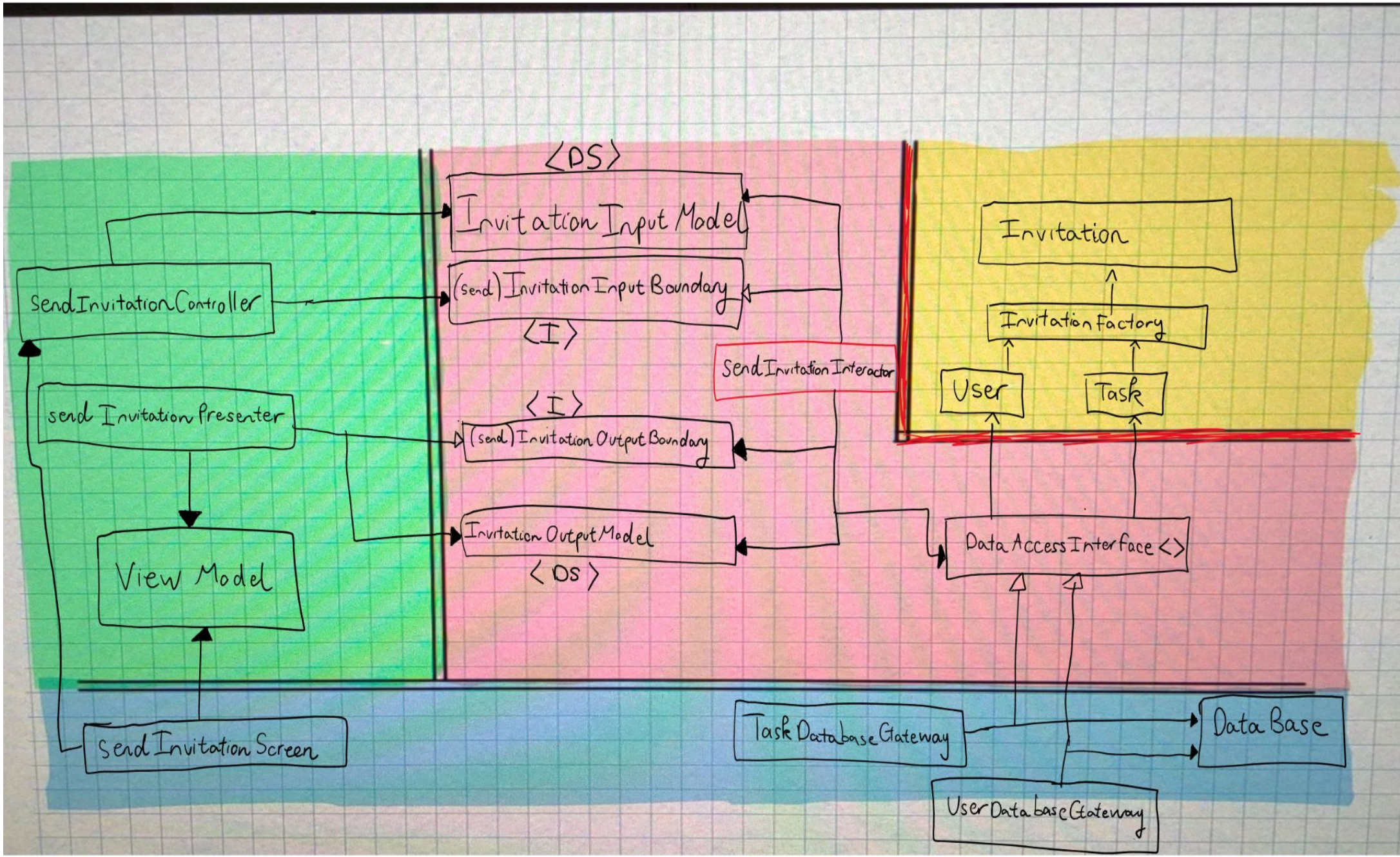
I feel much more confident in my ability to write clean code and take up more complex projects in the future.

Excited to work on more projects!

Lesson: Don't be scared to try new projects on your own. It is not as hard as it looks.

Image source: https://towardsdatascience.com/taking-the-plunge-into-data-science-my-experience-on-the-immersive-course-6e06b5b0e498

Send invitation use case



Invitation Management System Demo

to conclude…

thanks, Derek <3